

CSS532 HW2 Report

Name: Iris(Qiaoyu) Zheng

1 Feature Realization :

a). IoTConfigure your laptop as an IoT device, which can communicate with Azure IoT Hub

Create Azure IoT Hub

Dashboard > New > IoT Hub > IoT hub

IoT hub

Microsoft

Basics

Size and scale

Review + create

Create an IoT Hub to help you connect, monitor, and manage billions of your IoT assets. [Learn More](#)

PROJECT DETAILS

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ

Free Trial

Resource Group * ⓘ

Select existing...

Create new

Region * ⓘ

West US

IoT Hub Name * ⓘ

Name your IoT Hub

Dashboard > IrisIoTHub

IrisIoTHub

IoT Hub

Search (Cmd+J)

Move

Delete

Refresh

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Resource group (change) : IoT_Assignment

Status : Active

Current location : West US

Subscription (change) : Free Trial

Subscription ID : 516fdd7d-1a7d-4f5c-8ce4-8d47409f9c3c

Tags (change) : Click here to add tags

Hostname : IrisIoTHub.azure-devices.net

Pricing and scale tier : B1 - Basic

Number of IoT Hub units : 1

Upgrade?

Add new devices onto IoT hub

+ New

Refresh

Delete

View, create, delete, and update devices in your IoT Hub.

+ ×

Field

Operator

Value

select or enter a property name

=

specify constraint value

+ Add a new clause

Query devices

</> Switch to query editor

DEVICE ID	STATUS	LAST ACTIVITY TIME (UTC)	LAST STATUS UPDATE (UTC)	AUTHENTICATION TYPE	CLOUD T...
IrisLaptop	Enabled	Oct 23, 2019 5:39 AM	--	Sas	0

Create a local python file, get connection string and paste the string to the python as a constant value.

Successfully connected the laptop to the Azure IoT Hub with Azure IoT SDK.

```
# The connection string for a device should never be stored in code. For the sake of simplicity we're using ar
conn_str = os.getenv("IOTHUB_DEVICE_CONNECTION_STRING")
device_id = "IrisLaptop"
host_name = "IrisIoTHub"
# The client object is used to interact with your Azure IoT hub.
device_client = IoTHubDeviceClient.create_from_connection_string("HostName=IrisIoTHub.azure-devices.net;Device
# Connect the client.
device_client.connect()
```

b). Configure Azure IoT Hub to receive messages from your device (laptop) and show the result in Azure IoT Hub console

Do the configuration in python to connect the laptop to the console and send the message.

Python code:

```
# The connection string for a device should never be stored in code. For the sake of simplicity we're using an
conn_str = os.getenv("IOTHUB_DEVICE_CONNECTION_STRING")
device_id = "IrisLaptop"
host_name = "IrisIoTHub"
# The client object is used to interact with your Azure IoT hub.
device_client = IoTHubDeviceClient.create_from_connection_string("HostName=IrisIoTHub.azure-devices.net;DeviceI
# Connect the client.
device_client.connect()

message2 = {}
message2['data1'] = random.randrange(10)
message2['data2'] = random.randrange(10)
message2['data3'] = random.randrange(10)

messageJson2 = json.dumps(message2)
device_client.send_message(messageJson2)
time.sleep(1)

device_client.disconnect()
```

Monitor the message received on the IoT Console via iot hub monitor events:

```
[(base) Iris-MacBook-Pro:Assignment2 iriszheng$ az iot hub monitor-events --hub-name IrisIoTHub --device-id IrisLaptop
Starting event monitor, filtering on device: IrisLaptop, use ctrl-c to stop...
{
  "event": {
    "origin": "IrisLaptop",
    "payload": {
      "data1": 8,
      "data2": 2,
      "data3": 3
    }
  }
}
```

c). Connect Azure Stream Analytics to your Azure IoT Hub to process received messages from devices and save the raw data (the data shall be abstracted from received messages) and the processed data (you should use Stream Analytics to process the data) into Azure Blob.

Create a stream analytic job

The screenshot shows the 'ListenToIris' Stream Analytics job in the Azure portal. The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, and Locks. The main area displays job details: Resource group (IoT_Assignment), Status (Running), Location (West US), Subscription (Free Trial), and Subscription ID (516fdd7d-1a7d-4f5c-8ce4-8d47409f9c3c). On the right, there are links for Send feedback, UserVoice, and a table with job metadata: Created (Monday, October 21, 2019, 11:22:11 PM), Started (Tuesday, October 22, 2019, 10:38:45 PM), Output watermark (Tuesday, October 22, 2019, 11:44:04 PM), and Hosting environment (Cloud).

Add an input from IoT hub, set access policy name/key/ consumer group. Etc.

The screenshot shows the 'ListenToIris - Inputs' configuration page in the Azure portal. The left sidebar contains navigation links: Dashboard > All resources > ListenToIris - Inputs, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Settings, Locks, Job topology, Inputs, Functions, Query, Outputs, Configure, and Storage account settings. The main area displays a table with input details: Name (inputfromiot) and Source type (Stream). On the right, there is a 'Input details' section with a warning message: 'Inputs can't be added or edited while a job is running. You can stop the job to add or edit inputs.' Below the warning, there are fields for Subscription (Subscription information not needed), IoT Hub (IrisIoTHub), Endpoint (Messaging), Shared access policy name (iothubowner), Shared access policy key (*****), and Consumer group.

Create a storage account for storing the data.

The screenshot shows the Azure portal interface for a storage account named 'iotlaptopiris8217'. The left sidebar contains navigation options like 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Data transfer', 'Storage Explorer (preview)', and 'Settings'. The main area displays the account's configuration details, including the resource group 'IoT_Assignment', status 'Primary: Available', location 'Central US', and subscription ID '516fdd7d-1a7d-4f5c-8ce4-8d47409f9c3c'. The 'Performance' is set to 'Standard', 'Replication' is 'Locally-redundant storage (LRS)', and 'Account kind' is 'Storage (general purpose v1)'. Below the configuration, there are three service tiles: 'Containers' (REST-based object storage), 'File shares' (Serverless SMB file shares), and 'Tables' (Tabular data storage), each with a 'Learn more' link.

Add 2 blob storage output in the analytic job, one for store the raw data, and another for store processes data(get an average of the 3 data generated in this case). The set the output storing the data to the storage account just created.

The screenshot shows the 'ListenToIris - Outputs' configuration page for a Stream Analytics job. The left sidebar has options for 'Overview', 'Activity log', and 'Access control (IAM)'. The main area features a table with two columns: 'Name' and 'Sink'. There are two rows of output configurations:

Name	Sink
iotoutput	Blob storage
outputAverage	Blob storage

This is a configuration form for adding a new output. It includes a 'Subscription' dropdown with the text 'Subscription information not needed'. Below it is a 'Storage account' field with a red asterisk and an information icon, containing the text 'iotlaptopiris8217'. At the bottom is a 'Storage account key' field with a masked key represented by dots.

Edit Query doc, store the raw data to the blob, also, get the average of the data received and store the average as a newly processed data two another blob.

The screenshot shows the Azure Data Studio interface. On the left, the 'Inputs (1)' section contains 'inputfromiot'. The 'Outputs (4)' section contains 'iotoutput', 'outputAverage', 'outputprocess', and 'outputtoblob'. The main query editor displays the following SQL code:

```
1 SELECT
2   data1,
3   data2,
4   data3,
5   (data1+data2+data3)/3 AS data_average
6 INTO
7   iotoutput
8 FROM
9   inputfromiot
10
11 SELECT
12   (data1+data2+data3)/3 AS data_average
13 INTO
14   outputAverage
15 FROM
16   inputfromiot
17
```

Get the raw data:

The screenshot shows a JSON file named '0_954b6de8aa494fdb8e8ce0771d3dd2ff_1 (1).json' in a code editor. The file contains an array of four JSON objects, each representing a data record with fields 'data1', 'data2', 'data3', and 'data_average'.

```
1 [{"data1":1,"data2":1,"data3":2,"data_average":1}]
2 [{"data1":2,"data2":4,"data3":1,"data_average":2}]
3 [{"data1":5,"data2":1,"data3":7,"data_average":4}]
4 [{"data1":8,"data2":2,"data3":3,"data_average":4}]
```

The screenshot shows the Azure Blob Storage interface for a blob named '0_954b6de8aa494fdb8e8ce0771d3dd2ff_1.json'. The blob is 198 B in size and is stored in the 'application/octet-stream' format. The interface includes a toolbar with buttons for 'Save', 'Discard', 'Download', 'Refresh', 'Delete', 'Change tier', 'Acquire lease', and 'Break lease'. Below the toolbar, the blob's properties are listed:

Property	Value
SIZE	198 B
ACCESS TIER	N/A
ACCESS TIER LAST MODIFIED	N/A
SERVER ENCRYPTED	true
ETAG	0x8D7578403DA6849
CONTENT-TYPE	application/octet-stream
CONTENT-MD5	-
LEASE STATUS	Unlocked
LEASE STATE	Available
LEASE DURATION	-

Get the processed data

The screenshot shows a JSON file named '0_db547040cca940daad7f7c27521e73fc_1.json' in a code editor. The file contains an array of four JSON objects, each representing a processed data record with the field 'data_average'.

```
1 [{"data_average":1}]
2 [{"data_average":2}]
3 [{"data_average":4}]
4 [{"data_average":4}]
```

Dashboard > iotlaptopiris8217 - Containers > azure-webjobs-hosts > dataaverage/0_db547040cca940daad7f7c27521e73fc_1.json

dataaverage/0_db547040cca940daad7f7c27521e73fc_1.json
Blob

Save Discard Download Refresh Delete Change tier Acquire lease Break lease

Overview Snapshots Edit blob Generate SAS

Properties

URL	https://iotlaptopiris8217.blob.core.windows.net/dataaverage/0_db547040cca940daad7f7c27521e73fc_1.json
LAST MODIFIED	10/22/2019, 11:41:20 PM
CREATION TIME	10/22/2019, 10:39:59 PM
TYPE	Block blob
SIZE	78 B
ACCESS TIER	N/A
ACCESS TIER LAST MODIFIED	N/A

d). Connect Azure Functions to respond to Azure Blob events when new "processed data" is added. Your Azure Function may simply print out the content of newly added data, which is the "processed data" obtained in Step 3.

Create a new function app, set the correct language and consumer group (use node.js in this case)

Function App [Documentation](#)

UW

+ Add Edit columns Refresh Assign tags Start Restart Stop Delete

Subscriptions: Free Trial

Filter by name... All resource groups All locations All tags No grouping

3 items

<input type="checkbox"/>	Name ↑↓	Status	App Type	App Service Plan	Location ↑↓	Subscription ↑↓
<input type="checkbox"/>	iotlaptopiris	Stopped	Function App	CentralUSPlan	Central US	Free Trial

Create a new function, use existed azure blob storage trigger template here

All subscriptions

Function Apps

newDataProcess

Functions

BlobTrigger1

Integrate

Manage

Monitor

BlobTrigger2

Azure Blob Storage trigger

A function that will be run whenever a blob is added to a specified container

Binding trigger variables on the portal page can also do it in the JSON file.

Azure Blob Storage trigger [x delete](#)

Blob parameter name ⓘ

myBlob

Path ⓘ

azure-webjobs-hosts/dataaverage/0_e01f6bfa3acc437

Storage account connection ⓘ

[show value](#)

iotlaptopiris8217_STORAGE

[new](#)

Run node.js code that prints the processed data whenever the blob receives a new message.

index.js

Save

Run

```
1 module.exports = async function (context, myBlob) {
2   context.log("JavaScript blob trigger function processed blob \n Name:", context.bindingData.name, "\n Blob
3   context.log(myBlob.toString());
4 };
```

```
2019-10-23T07:20:45.640 [Information] Executing 'Functions.BlobTrigger2' (Reason='New blob detected: azure-webjobs-
hosts/dataaverage/0_e01f6bfa3acc4374b456802d7a38e2eb_1.json', Id=d0897dcc-94eb-405e-abb9-5e059fc7d48d)
2019-10-23T07:20:45.647 [Information] JavaScript blob trigger function processed blob
Name: undefined
Blob Size: 58 Bytes
2019-10-23T07:20:45.647 [Information] {"data_average":3}
{"data_average":5}
{"data_average":3}
```

```
2019-10-23T07:21:37.097 [Information] Executing 'Functions.BlobTrigger2' (Reason='New blob detected: azure-webjobs-
hosts/dataaverage/0_e01f6bfa3acc4374b456802d7a38e2eb_1.json', Id=c59f0aea-51a1-413a-9632-927acbeaf2e3)
2019-10-23T07:21:37.107 [Information] JavaScript blob trigger function processed blob
Name: undefined
Blob Size: 78 Bytes
2019-10-23T07:21:37.107 [Information] {"data_average":3}
{"data_average":5}
{"data_average":3}
{"data_average":4}
2019-10-23T07:21:37.108 [Information] Executed 'Functions.BlobTrigger2' (Succeeded, Id=c59f0aea-51a1-413a-9632-
927acbeaf2e3)
```

```
2019-10-23T07:22:28.343 [Information] JavaScript blob trigger function processed blob
Name: undefined
Blob Size: 98 Bytes
2019-10-23T07:22:28.343 [Information] {"data_average":3}
{"data_average":5}
{"data_average":3}
{"data_average":4}
{"data_average":4}
2019-10-23T07:22:28.344 [Information] Executed 'Functions.BlobTrigger2' (Succeeded, Id=181b2d69-2c36-4d44-a783-
6ce8bclf29ae)
```

2 Problems Solving :

a). Select data in Query

Select * didn't work for some reason, then I fixed it by selecting specific data, but still wanna know the reason.

b). Storage Limitation

Now in my blob, a new JSON file will be created automatically when the it reaches a specific size, didn't find a place to do the setting.

c). Make deployment working on Visual Studio

I did the task4 on Visual Studio originally but met several problems, as .net doesn't work well when deploying the function. Then I went back to the portal page and used the binding extensions provided on the portal, which works well.

3 Time Consuming

Around 8 hours